

**1. (18 points)**

**a. (13 points)**

Using Project 1 Relations/Tables (**including VIEWS**) create them in the Oracle DBMS database. Create **Project 2.sql** that contains the Commented SQL commands.

In **Project 2.doc** run one command at the time and take a screenshot of the commented command and its output and insert them in this document.

**TABLE CUSTOMER**

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a connection icon labeled 'PS-3380'. Below the toolbar, there are tabs for 'Worksheet' and 'Query Builder', with 'Worksheet' currently selected. The main workspace contains the following SQL code:

```
/* DELETING CUSTOMER TABLE IF ANY EXISTS EARLIER */
DROP TABLE CUSTOMER CASCADE CONSTRAINTS;

/* CREATING TABLE CUSTOMER */
CREATE TABLE CUSTOMER (CUSTOMERID NUMBER,
                      CUSTOMERNAME VARCHAR2(50),
                      CUSTOMERADDRESS VARCHAR2(100),
                      CUSTOMERCITY VARCHAR2(20),
                      CUSTOMERSTATE VARCHAR2(15),
                      CUSTOMERPOSTALCODE VARCHAR2(15),
                      CUSTOMEREMAIL VARCHAR2(30),
                      CUSTOMERUSERNAME VARCHAR2(20),
                      CUSTOMERPASSWORD VARCHAR2(30),
                      PRIMARY KEY (CUSTOMERID));
```

Below the workspace, there are two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, showing the output of the executed commands:

```
Table CUSTOMER dropped.

Table CUSTOMER created.
```

## TABLE TERRITORY

The screenshot shows the SQL Server Management Studio interface. In the top window, titled 'PS-3380', there is a 'Worksheet' tab and a 'Query Builder' tab. The 'Query Builder' tab contains the following T-SQL script:

```
/* DELETING TERRITORY TABLE IF ANY EXISTS EARLIER */
DROP TABLE TERRITORY CASCADE CONSTRAINTS;

/* CREATING TABLE TERRITORY */
CREATE TABLE TERRITORY( TERRITORYID NUMBER,
                        TERRITORYNAME VARCHAR2 (30),
                        PRIMARY KEY (TERRITORYID));
```

In the bottom window, titled 'Script Output' and 'Query Result', it shows the results of the task completion:

```
Table TERRITORY dropped.
```

Table TERRITORY created.

## TABLE SALESPERSON

The screenshot shows the SQL Server Management Studio interface. In the top window, titled 'PS-3380', there is a 'Worksheet' tab and a 'Query Builder' tab. The 'Query Builder' tab contains the following T-SQL script:

```
/* DELETING SALESPERSON TABLE IF ANY EXISTS EARLIER */
DROP TABLE SALESPERSON CASCADE CONSTRAINTS;

/* CREATING TABLE SALESPERSON */
CREATE TABLE SALESPERSON ( SALESPERSONID NUMBER,
                           SALESPERSONNAME VARCHAR2 (30),
                           SALESPERSONPHONE VARCHAR2 (15),
                           SALESPERSONEMAIL VARCHAR2 (100),
                           SALESPERSONUSERNAME VARCHAR2 (50),
                           SALESPERSONPASSWORD VARCHAR2 (50),
                           TERRITORYID NUMBER,
                           PRIMARY KEY (SALESPERSONID),
                           FOREIGN KEY (TERRITORYID) REFERENCES TERRITORY (TERRITORYID) ON DELETE CASCADE);
```

In the bottom window, titled 'Script Output' and 'Query Result', it shows the results of the task completion:

```
Table SALESPERSON dropped.
```

Table SALESPERSON created.

## TABLE DOESBUSINESSIN

The screenshot shows the SQL Server Management Studio interface with a query window titled 'Worksheet' containing the following SQL code:

```
/* DELETING DOESBUSINESSIN TABLE IF ANY EXISTS EARLIER */
DROP TABLE DOESBUSINESSIN CASCADE CONSTRAINTS;

/* CREATING TABLE DOESBUSINESS */
CREATE TABLE DOESBUSINESSIN (
    CUSTOMERID NUMBER,
    TERRITORYID NUMBER,
    PRIMARY KEY (CUSTOMERID, TERRITORYID),
    FOREIGN KEY (CUSTOMERID) REFERENCES CUSTOMER (CUSTOMERID) ON DELETE CASCADE,
    FOREIGN KEY (TERRITORYID) REFERENCES TERRITORY (TERRITORYID) ON DELETE CASCADE);
```

The status bar at the bottom indicates 'Task completed in 1.704 seconds'. Below the query window, the output pane shows the results of the table creation.

```
Table DOESBUSINESSIN dropped.

Table DOESBUSINESSIN created.
```

## TABLE PRODUCTLINE

The screenshot shows the SQL Server Management Studio interface with a query window titled 'Worksheet' containing the following SQL code:

```
/* DELETING PRODUCTLINE TABLE IF ANY EXISTS EARLIER */
DROP TABLE PRODUCTLINE CASCADE CONSTRAINTS;

/* CREATING TABLE PRODUCTLINE */
CREATE TABLE PRODUCTLINE (
    PRODUCTLINEID NUMBER,
    PRODUCTLINENAME VARCHAR2(40),
    PRIMARY KEY (PRODUCTLINEID));
```

The status bar at the bottom indicates 'Task completed in 0.7 seconds'. Below the query window, the output pane shows the results of the table creation.

```
Table PRODUCTLINE dropped.

Table PRODUCTLINE created.
```

## TABLE PRODUCT

The screenshot shows the Oracle SQL Developer interface with the following details:

- Toolbar:** Includes icons for connection, script, query, search, and various tools.
- Top Status Bar:** Shows the connection name "PS-3380 X" and the time "0.454 seconds".
- Worksheet Tab:** Active tab, showing the SQL code for creating the PRODUCT table.
- SQL Code:**

```
DROP TABLE PRODUCT CASCADE CONSTRAINTS;

/* CREATING TABLE PRODUCT */
CREATE TABLE PRODUCT (
    PRODUCTID NUMBER,
    PRODUCTNAME VARCHAR2(30),
    PRODUCTFINISH VARCHAR2(30),
    PRODUCTSTANDARDPRICE NUMBER,
    PRODUCTLINEID NUMBER,
    PHOTO VARCHAR2(20),
    PRIMARY KEY (PRODUCTID),
    FOREIGN KEY (PRODUCTLINEID) REFERENCES PRODUCTLINE(PRODUCTLINEID) ON DELETE CASCADE);
```
- Script Output Tab:** Shows the message "Table PRODUCT dropped.".
- Query Result Tab:** Shows the message "Table PRODUCT created.".
- Bottom Status Bar:** Shows the message "Task completed in 0.454 seconds".

## TABLE ORDERS

The screenshot shows the Oracle SQL Developer interface with the following details:

- Toolbar:** Includes icons for connection, script, query, search, and various tools.
- Top Status Bar:** Shows the connection name "PS-3380 X" and the time "0.69 seconds".
- Worksheet Tab:** Active tab, showing the SQL code for creating the ORDERS table.
- SQL Code:**

```
DROP TABLE ORDERS CASCADE CONSTRAINTS;

/* CREATING TABLE ORDERS */
CREATE TABLE ORDERS (
    ORDERID NUMBER,
    ORDERDATE DATE,
    CUSTOMERID NUMBER,
    PRIMARY KEY (ORDERID),
    FOREIGN KEY (CUSTOMERID) REFERENCES CUSTOMER(CUSTOMERID) ON DELETE CASCADE);
```
- Script Output Tab:** Shows the message "Table ORDERS dropped.".
- Query Result Tab:** Shows the message "Table ORDERS created.".
- Bottom Status Bar:** Shows the message "Task completed in 0.69 seconds".

## TABLE ORDERLINE

The screenshot shows a SQL Workbench interface with a worksheet tab active. The code in the worksheet creates the ORDERLINE table with columns ORDERID, PRODUCTID, ORDEREDQUANTITY, and SALEPRICE, with primary key constraints on ORDERID and PRODUCTID, and foreign key constraints referencing ORDERS and PRODUCT tables respectively.

```
/* DELETING ORDERLINE TABLE IF ANY EXISTS EARLIER */
DROP TABLE ORDERLINE CASCADE CONSTRAINTS;
/* CREATING TABLE ORDERLINE */
CREATE TABLE ORDERLINE(
    ORDERID NUMBER,
    PRODUCTID NUMBER,
    ORDEREDQUANTITY NUMBER,
    SALEPRICE NUMBER,
    PRIMARY KEY (ORDERID, PRODUCTID),
    FOREIGN KEY (ORDERID) REFERENCES ORDERS(ORDERID) ON DELETE CASCADE,
    FOREIGN KEY (PRODUCTID) REFERENCES PRODUCT(PRODUCTID) ON DELETE CASCADE);
```

Script Output X | Query Result X  
Task completed in 0.737 seconds

Table ORDERLINE dropped.

Table ORDERLINE created.

## TABLE PRICEUPDATE

The screenshot shows a SQL Workbench interface with a worksheet tab active. The code in the worksheet creates the PRICEUPDATE table with columns PRICEUPDATEID, DATECHANGED, OLDPRIICE, and NEWPRICE, with a primary key constraint on PRICEUPDATEID.

```
/* DELETING PRICEUPDATE TABLE IF ANY EXISTS EARLIER */
DROP TABLE PRICEUPDATE CASCADE CONSTRAINTS;

/* CREATING TABLE PRICEUPDATE */
CREATE TABLE PRICEUPDATE(
    PRICEUPDATEID NUMBER,
    DATECHANGED DATE,
    OLDPRIICE NUMBER,
    NEWPRICE NUMBER,
    PRIMARY KEY (PRICEUPDATEID));
```

Script Output X | Query Result X  
Task completed in 0.58 seconds

Table PRICEUPDATE dropped.

Table PRICEUPDATE created.

## VIEW:

### VIEW COMPARISON

The screenshot shows a SQL Server Management Studio window with the title bar "PS-3380" and "COMPARISON". The toolbar includes standard icons for file operations, zoom, and search. The status bar at the bottom indicates "0.26100001 seconds". The main area is a "Query Builder" worksheet tab. The code in the query editor is as follows:

```
/* DELETE OR REPLACE VIEW COMPARISON IF ANY EXISTS EARLIER */
DROP VIEW COMPARISON CASCADE CONSTRAINTS;

/* CREATING OR REPLACING VIEW COMPARISON */
CREATE OR REPLACE VIEW COMPARISON(PRODUCTLINEID, PRODUCTID, PRODUCTNAME, SALESPRICE, ORDEREDQUANTITY, TOTALSALES) AS
SELECT PL.PRODUCTLINEID, P.PRODUCTID, P.PRODUCTNAME, OL.SALESPRICE, SUM(OL.ORDEREDQUANTITY),
(SUM(OL.ORDEREDQUANTITY)*OL.SALEPRICE)
FROM PRODUCTLINE PL, PRODUCT P, ORDERLINE OL
WHERE PL.PRODUCTLINEID = P.PRODUCTLINEID AND P.PRODUCTID = OL.PRODUCTID
GROUP BY PL.PRODUCTLINEID, P.PRODUCTID, P.PRODUCTNAME, OL.SALESPRICE
ORDER BY PL.PRODUCTLINEID;
```

Below the query editor, there are two tabs: "Script Output" and "Query Result". The status bar at the bottom of the window says "Task completed in 0.261 seconds".

View COMPARISON dropped.

View COMPARISON created.

### VIEW TOTALVALUE

The screenshot shows a SQL Server Management Studio window with the title bar "PS-3380" and "COMPARISON". The toolbar and status bar are identical to the previous screenshot. The main area is a "Query Builder" worksheet tab. The code in the query editor is as follows:

```
/* DELETE OR REPLACE VIEW TOTALVALUE IF ANY EXISTS EARLIER */
DROP VIEW TOTALVALUE CASCADE CONSTRAINTS;

/* CREATING OR REPLACING VIEW TOTALVALUE */
CREATE OR REPLACE VIEW TOTALVALUE(PRODUCTID, PRODUCTNAME, SALESPRICE, ORDEREDQUANTITY, TOTALSALES) AS
SELECT P.PRODUCTID, P.PRODUCTNAME, OL.SALESPRICE, SUM(OL.ORDEREDQUANTITY), (SUM(OL.ORDEREDQUANTITY)*OL.SALEPRICE)
FROM PRODUCT P, ORDERLINE OL
WHERE P.PRODUCTID = OL.PRODUCTID
GROUP BY P.PRODUCTID, P.PRODUCTNAME, OL.SALESPRICE
ORDER BY P.PRODUCTID;
```

Below the query editor, there are two tabs: "Script Output" and "Query Result". The status bar at the bottom of the window says "Task completed in 0.411 seconds".

View TOTALVALUE dropped.

View TOTALVALUE created.

## **VIEW TERRITORYSALE**

The screenshot shows the Oracle SQL Developer interface with the title bar "PS-3380 X COMPARISION X". The main area displays a SQL script for creating a view:

```
/* DELETE OR REPLACE VIEW TERRITORYSALE IF ANY EXISTES EARLIER */
DROP VIEW TERRITORYSALE CASCADE CONSTRAINTS;

/* CREATING OR REPLACING VIEW TERRITORYSALE */
CREATE OR REPLACE VIEW TERRITORYSALE(SALESPERSONID, TERRITORYID, CUSTOMERID, PRODUCTID, PROODUCTSATNDARDPRICE) AS
SELECT S.SALESPERSONID, B.TERRITORYID, C.CUSTOMERID, P.PRODUCTID, P.PRODUCTSTANDARDPRICE
FROM CUSTOMER C, PRODUCT P, SALESPERSON S, ORDERLINE OL, ORDERS O, DOESBUSINESSIN B
WHERE OL.PRODUCTID = P.PRODUCTID AND O.ORDERID = OL.ORDERID AND C.CUSTOMERID = O.CUSTOMERID
AND B.CUSTOMERID = C.CUSTOMERID AND B.TERRITORYID = S.TERRITORYID
ORDER BY S.SALESPERSONID;
```

Below the code, the status bar indicates "Task completed in 0.276 seconds". The output pane shows the message "View TERRITORYSALE dropped." followed by "View TERRITORYSALE created.".

## **VIEW SHIPMENT**

The screenshot shows the Oracle SQL Developer interface with the title bar "PS-3380 X COMPARISION X". The main area displays a SQL script for creating a view:

```
/* DELETE OR REPLACE VIEW SHIPMENT IF ANY EXISTES EARLIER */
DROP VIEW SHIPMENT CASCADE CONSTRAINTS;

/* CREATING OR REPLACING VIEW SHIPMENT */
CREATE OR REPLACE VIEW SHIPMENT(CUSTOMERSTATE, CUSTOMERCOUNT) AS
SELECT C.CUSTOMERSTATE, COUNT(C.CUSTOMERID)
FROM CUSTOMER C
GROUP BY CUSTOMERSTATE;
```

Below the code, the status bar indicates "Task completed in 0.235 seconds". The output pane shows the message "View SHIPMENT dropped." followed by "View SHIPMENT created.".

## VIEW PURCHASEHISTORY

The screenshot shows the Oracle SQL Developer interface with a query editor window titled 'PS-3380 X ORDERLINE X'. The code in the editor is:

```
/* DELETE OR REPLACE VIEW PURCHASEHISTORY IF ANY EXISTES EARLIER */
DROP VIEW PURCHASEHISTORY CASCADE CONSTRAINTS;

/* CREATING OR REPLACING VIEW PURCHASEHISTORY */
CREATE OR REPLACE VIEW PURCHASEHISTORY(CUSTOMERID, CUSTOMERNAME, ORDERID, ORDERDATE, ORDERQUANTITY, PRODUCTNAME,
PRODUCTSTANDARDPRICE)
AS SELECT C.CUSTOMERID, C.CUSTOMERNAME, O.ORDERID, O.ORDERDATE, OL.ORDEREDQUANTITY, P.PRODUCTNAME, P.PRODUCTSTANDARDPRICE
FROM CUSTOMER C, ORDERS O, ORDERLINE OL, PRODUCT P
WHERE C.CUSTOMERID = O.CUSTOMERID AND OL.ORDERID = O.ORDERID AND P.PRODUCTID = OL.PRODUCTID
ORDER BY C.CUSTOMERID;
```

Below the editor, there are tabs for 'Script Output' and 'Query Result'. A status bar at the bottom indicates 'Task completed in 0.272 seconds'.

Output messages in the log area:

- View PURCHASEHISTORY dropped.
- View PURCHASEHISTORY created.

## VIEW ORDERDETAIL

The screenshot shows the Oracle SQL Developer interface with a query editor window titled 'PS-3380 X ORDERLINE X'. The code in the editor is:

```
/* DELETE OR REPLACE VIEW ORDERDETAIL IF ANY EXISTES EARLIER */
DROP VIEW ORDERDETAIL CASCADE CONSTRAINTS;

/* CREATING OR REPLACING VIEW ORDERDETAIL */
CREATE OR REPLACE VIEW ORDERDETAIL(ORDERID, ORDERDATE, CUSTOMERID) AS
SELECT O.ORDERID, O.ORDERDATE, C.CUSTOMERID
FROM ORDERS O, CUSTOMER C
WHERE O.CUSTOMERID = C.CUSTOMERID;
```

Below the editor, there are tabs for 'Script Output' and 'Query Result'. A status bar at the bottom indicates 'Task completed in 0.197 seconds'.

Output messages in the log area:

- View ORDERDETAIL dropped.
- View ORDERDETAIL created.

## TRIGGER

The screenshot shows the Oracle SQL Developer interface with two main panes: a Worksheet pane and a Script Output pane.

**Worksheet Pane:** Contains the SQL code for creating a trigger named STANDARDPRICEUPDATE. The code is as follows:

```
/* TRIGGER FOR STANDARDPRICEUPDATE */
CREATE OR REPLACE TRIGGER STANDARDPRICEUPDATE
AFTER UPDATE OF PRODUCTSTANDARDPRICE ON PRODUCT
FOR EACH ROW
DECLARE
    CURRENTDATE DATE;
BEGIN
    SELECT CURRENTDATE INTO CURRENTDATE
    FROM SYS.DUAL;

    INSERT INTO PRICEUPDATE (PRICEUPDATEID, DATECHANGED, OLDPRI  
CE, NEWPRICE)
    VALUES (:OLD.PRODUCTSTANDARDPRICE, :NEW.PRODUCTSTANDARDPRICE);
    DBMS_OUTPUT.PUT_LINE('SUCCESSFULLY ADDED');

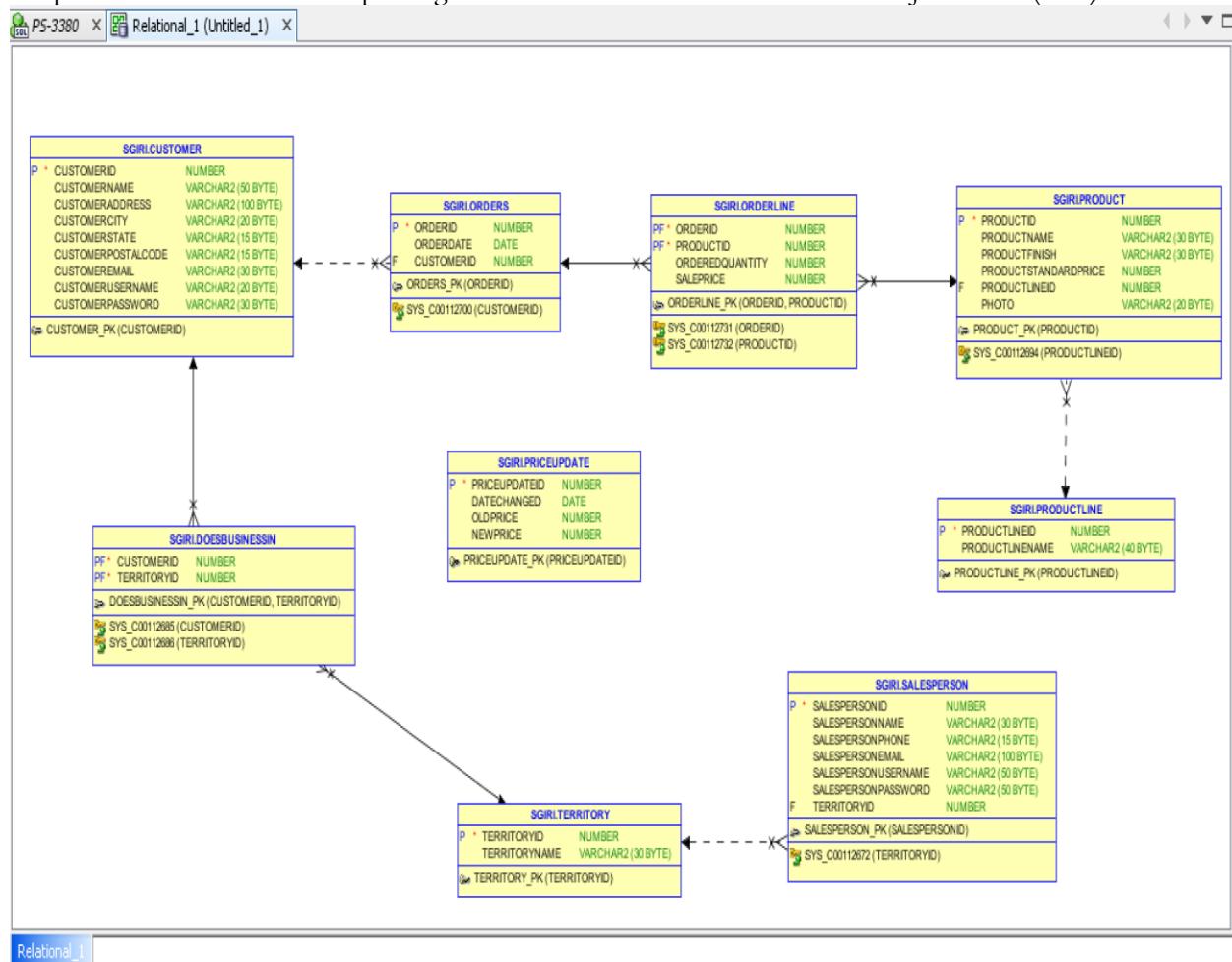
END;
```

**Script Output Pane:** Shows the result of the trigger compilation. It displays the message "Trigger STANDARDPRICEUPDATE compiled" and indicates the task completed in 0.224 seconds.

```
Trigger STANDARDPRICEUPDATE compiled
Task completed in 0.224 seconds
```

b. (5 points)

Request the Table Relationships Diagram for the **Data**base and attach it in Project 2.doc (here).



2. [40 points]

The Client has supplied the following data. Use **SQL** to **insert these data** in the Oracle DBMS database created in **Project 1**.

## Customer

CustomerID, CustomerName, CustomerAddress, CustomerCity, CustomerState, CustomerPostalCode, CustomerEmail, CustomerUserName, CustomerPassword

1, 'Contemporary Casuals', '1355 S Hines Blvd', 'Gainesville', 'FL', '32601-2871'

2, 'Value Furnitures', '15145 S.W. 17th St.', 'Plano', 'TX', '75094-7734'

3, 'Home Furnishings', '1900 Allard Ave', 'Albany', 'NY', '12209-1125',

**'homefurnishings?@gmail.com', 'CUSTOMER1', 'CUSTOMER1#'**

4, 'Eastern Furniture', '1925 Beltline Rd.', 'Carteret', 'NJ', '07008-3188'

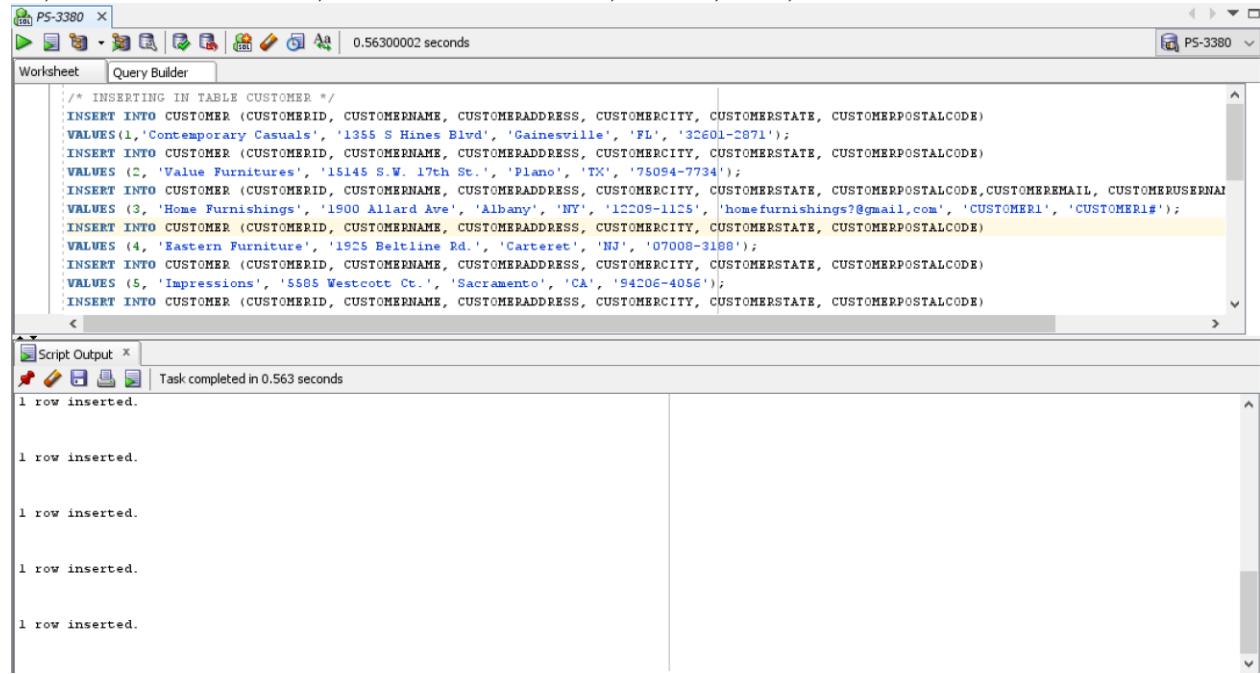
5, 'Impressions', '5585 Westcott Ct.', 'Sacramento', 'CA', '94206-4056'

6, 'Furniture Gallery', '325 Flatiron Dr.', 'Boulder', 'CO', '80514-4432'

7, 'New Furniture', 'Palace Ave', 'Farmington', 'NM', ''

8, 'Dunkins Furniture', '7700 Main St', 'Syracuse', 'NY', '31590'

- 9, 'A Carpet', '434 Abe Dr', 'Rome', 'NY', '13440'  
**12**, 'Flanigan Furniture', 'Snow Flake Rd', 'Ft Walton Beach', 'FL', '32548'  
13, 'Ikards', '1011 S. Main St', 'Las Cruces', 'NM', '88001'  
14, 'Wild Bills', 'Four Horse Rd', 'Oak Brook', 'IL', '60522'  
15, 'Janet's Collection', 'Janet Lane', 'Virginia Beach', 'VA', '10012'  
16, 'ABC Furniture Co.', '152 Geramino Drive', 'Rome', 'NY', '13440'



The screenshot shows a database interface with two tabs: 'Worksheet' and 'Script Output'. The 'Worksheet' tab contains an SQL script for inserting data into the 'CUSTOMER' table. The 'Script Output' tab shows the results of the execution, indicating 1 row inserted for each of the 16 entries listed above.

```

/*
 * INSERTING IN TABLE CUSTOMER */
INSERT INTO CUSTOMER (CUSTOMERID, CUSTOMERNAME, CUSTOMERADDRESS, CUSTOMERCITY, CUSTOMERSTATE, CUSTOMERPOSTALCODE)
VALUES (1, 'Contemporary Casuals', '1355 S Hines Blvd', 'Gainesville', 'FL', '32601-2871');
INSERT INTO CUSTOMER (CUSTOMERID, CUSTOMERNAME, CUSTOMERADDRESS, CUSTOMERCITY, CUSTOMERSTATE, CUSTOMERPOSTALCODE)
VALUES (2, 'Value Furnitures', '15145 S.W. 17th St.', 'Plano', 'TX', '75084-7734');
INSERT INTO CUSTOMER (CUSTOMERID, CUSTOMERNAME, CUSTOMERADDRESS, CUSTOMERCITY, CUSTOMERSTATE, CUSTOMERPOSTALCODE, CUSTOMEREMAIL, CUSTOMERUSERNA
VALUES (3, 'Home Furnishings', '1900 Allard Ave', 'Albany', 'NY', '12208-1125', 'homefurnishings@gmail.com', 'CUSTOMER1', 'CUSTOMER1#');
INSERT INTO CUSTOMER (CUSTOMERID, CUSTOMERNAME, CUSTOMERADDRESS, CUSTOMERCITY, CUSTOMERSTATE, CUSTOMERPOSTALCODE)
VALUES (4, 'Eastern Furniture', '1925 Beltline Rd.', 'Carteret', 'NJ', '07008-3188');
INSERT INTO CUSTOMER (CUSTOMERID, CUSTOMERNAME, CUSTOMERADDRESS, CUSTOMERCITY, CUSTOMERSTATE, CUSTOMERPOSTALCODE)
VALUES (5, 'Impressions', '5505 Westcott Ct.', 'Sacramento', 'CA', '94206-4056');
INSERT INTO CUSTOMER (CUSTOMERID, CUSTOMERNAME, CUSTOMERADDRESS, CUSTOMERCITY, CUSTOMERSTATE, CUSTOMERPOSTALCODE)

```

Script Output:

```

1 row inserted.

```

## Salesperson

- SalespersonID, SalespersonName, SalespersonPhone, SalespersonEmail, SalespersonUserName,  
SalespersonPassword, TerritoryID
- 1, 'Doug Henny', '8134445555', '**'salesperson?@gmail.com'**', '**'SALESPERSON'**',  
**'SALESPERSON#'**, 1  
2, 'Robert Lewis', '8139264006', "", "", 2  
3, 'William Strong', '5053821212', "", "", 3  
4, 'Julie Dawson', '4355346677', "", "", 4  
5, 'Jacob Winslow', '2238973498', "", "", 5

The screenshot shows the Oracle SQL Developer interface. The top window is titled 'PS-3380' and contains a 'Worksheet' tab with a query builder. The query is:

```

/* INSERTING IN TABLE SALESPERSON */
INSERT INTO SALESPERSON (SALESPERSONID, SALESPERSONNAME, SALESPERSONPHONE, SALESPERSONEMAIL, SALESPERSONUSERNAME, SALESPERSONPASSWORD, TERRITORYID)
VALUES ('1', 'Doug Henny', '8134445555', 'salesperson@gmail.com', 'SALESPERSON', 'SALESPERSON#', '1');
INSERT INTO SALESPERSON (SALESPERSONID, SALESPERSONNAME, SALESPERSONPHONE, TERRITORYID)
VALUES ('2', 'Robert Lewis', '8139264006', '2');
INSERT INTO SALESPERSON (SALESPERSONID, SALESPERSONNAME, SALESPERSONPHONE, TERRITORYID)
VALUES ('3', 'William Strong', '5053821212', '3');
INSERT INTO SALESPERSON (SALESPERSONID, SALESPERSONNAME, SALESPERSONPHONE, TERRITORYID)
VALUES ('4', 'Julie Dawson', '4355346677', '4');
INSERT INTO SALESPERSON (SALESPERSONID, SALESPERSONNAME, SALESPERSONPHONE, TERRITORYID)
VALUES ('5', 'Jacob Winslow', '2238973458', '5');

```

The bottom window is titled 'Script Output' and shows the results of the execution:

```

Script Output X
Task completed in 0.264 seconds
1 row inserted.

```

## Territory

TerritoryID, TerritoryName

- 1, 'SouthEast'
- 2, 'SouthWest'
- 3, 'NorthEast'
- 4, 'NorthWest'
- 5, 'Central'

The screenshot shows a SQL Server Management Studio window with two panes. The top pane, titled 'Worksheet', contains a script of SQL INSERT statements for the 'TERRITORY' table. The bottom pane, titled 'Script Output', shows the results of the execution.

```
/* INSERTING IN TABLE TERRITORY */
INSERT INTO TERRITORY(TERRITORYID, TERRITORYNAME)
VALUES ('1', 'SOUTHEAST');
INSERT INTO TERRITORY(TERRITORYID, TERRITORYNAME)
VALUES ('2', 'SOUTHWEST');
INSERT INTO TERRITORY(TERRITORYID, TERRITORYNAME)
VALUES ('3', 'NORTHEAST');
INSERT INTO TERRITORY(TERRITORYID, TERRITORYNAME)
VALUES ('4', 'NORTHWEST');
INSERT INTO TERRITORY(TERRITORYID, TERRITORYNAME)
VALUES ('5', 'CENTRAL');
```

Script Output

```
Task completed in 0.328 seconds
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

**DoesBusinessIn**  
CustomerID, TerritoryID

1, 1  
2, 2  
3, 3  
4, 4  
5, 5  
6, 1  
7, 2

The screenshot shows the SQL Server Management Studio interface. The top window is titled 'PS-3380' and contains a query script in the 'Worksheet' tab:

```

INSERT INTO DOESBUSINESSIN (CUSTOMERID, TERRITORYID)
VALUES ('2', '2');
INSERT INTO DOESBUSINESSIN (CUSTOMERID, TERRITORYID)
VALUES ('3', '3');
INSERT INTO DOESBUSINESSIN (CUSTOMERID, TERRITORYID)
VALUES ('4', '4');
INSERT INTO DOESBUSINESSIN (CUSTOMERID, TERRITORYID)
VALUES ('5', '5');
INSERT INTO DOESBUSINESSIN (CUSTOMERID, TERRITORYID)
VALUES ('6', '1');
INSERT INTO DOESBUSINESSIN (CUSTOMERID, TERRITORYID)
VALUES ('7', '2');

```

The bottom window is titled 'Script Output' and shows the results of the execution:

```

1 row inserted.

```

The status bar at the bottom of the interface indicates 'Task completed in 0.391 seconds'.

## Product

ProductID, ProductName, ProductFinish, ProductStandardPrice, ProductLineID, **Photo**

- 1, 'End Table', 'Cherry', 175, 1, **table.jpg**
- 2, 'Coffee Table', 'Natural Ash', 200, 2
- 3, 'Computer Desk', 'Natural Ash', 375, 2
- 4, 'Entertainment Center', 'Natural Maple', 650, 3
- 5, 'Writers Desk', 'Cherry', 325, 1
- 6, '8-Drawer Desk', 'White Ash', 750, 2
- 7, 'Dining Table', 'Natural Ash', 800, 2
- 8, 'Computer Desk', 'Walnut', 250, 3

The screenshot shows a SQL database interface with two tabs: 'Worksheet' and 'Script Output'.  
In the 'Worksheet' tab, there is a code editor containing the following SQL script:

```
INSERT INTO PRODUCT (PRODUCTID, PRODUCTNAME, PRODUCTFINISH, PRODUCTSTANDARDPRICE, PRODUCTLINEID)
VALUES ('3', 'Computer Desk', 'Natural Ash', '375', '2');
INSERT INTO PRODUCT (PRODUCTID, PRODUCTNAME, PRODUCTFINISH, PRODUCTSTANDARDPRICE, PRODUCTLINEID)
VALUES ('4', 'Entertainment Center', 'Natural Maple', '650', '3');
INSERT INTO PRODUCT (PRODUCTID, PRODUCTNAME, PRODUCTFINISH, PRODUCTSTANDARDPRICE, PRODUCTLINEID)
VALUES ('5', 'Writers Desk', 'Cherry', '325', '1');
INSERT INTO PRODUCT (PRODUCTID, PRODUCTNAME, PRODUCTFINISH, PRODUCTSTANDARDPRICE, PRODUCTLINEID)
VALUES ('6', '8-Drawer Desk', 'White Ash', '750', '2');
INSERT INTO PRODUCT (PRODUCTID, PRODUCTNAME, PRODUCTFINISH, PRODUCTSTANDARDPRICE, PRODUCTLINEID)
VALUES ('7', 'Dining Table', 'Natural Ash', '800', '2');
INSERT INTO PRODUCT (PRODUCTID, PRODUCTNAME, PRODUCTFINISH, PRODUCTSTANDARDPRICE, PRODUCTLINEID)
VALUES ('8', 'Computer Desk', 'Walnut', '250', '3');
```

In the 'Script Output' tab, the results of the execution are shown:

```
Task completed in 0.316 seconds
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

## ProductLine

ProductLineID, ProductLineName

- 1, 'Cherry Tree'
- 2, 'Scandinavia'
- 3, 'Country Look'

The screenshot shows the SQL Server Management Studio interface. The top window is titled "PS-3380" and contains a query script for inserting data into the "PRODUCTLINE" table. The bottom window is titled "Script Output" and displays the results of the execution.

```
/* INSERTING IN TABLE PRODUCTLINE */
INSERT INTO PRODUCTLINE (PRODUCTLINEID, PRODUCTLINENAME)
VALUES ('1', 'Cheery Tree');
INSERT INTO PRODUCTLINE (PRODUCTLINEID, PRODUCTLINENAME)
VALUES ('2', 'Scandinavia');
INSERT INTO PRODUCTLINE (PRODUCTLINEID, PRODUCTLINENAME)
VALUES ('3', 'Country Look');
```

Task completed in 0.266 seconds

1 row inserted.

1 row inserted.

1 row inserted.

## Order

OrderID, OrderDate, CustomerID

1001, '21/Aug/16', 1

1002, '21/Jul/16', 8

1003, '22/ Aug/16', 15

1004, '22/Oct/16', 5

1005, '24/Jul/16', 3

1006, '24/Oct/16', 2

1007, '27/ Aug/16', 5

1008, '30/Oct/16', 12

1009, '05/Nov/16', 4

1010, '05/Nov/16', 1

PS-3380 X

0.46900001 seconds

Worksheet Query Builder

```
/* INSERTING IN TABLE ORDER */
INSERT INTO ORDERS (ORDERID, ORDERDATE, CUSTOMERID)
VALUES ('1001', TO_DATE('2016-08-21', 'YYYY-MM-DD'), '1');
INSERT INTO ORDERS (ORDERID, ORDERDATE, CUSTOMERID)
VALUES ('1002', TO_DATE('2016-07-21', 'YYYY-MM-DD'), '8');
INSERT INTO ORDERS (ORDERID, ORDERDATE, CUSTOMERID)
VALUES ('1003', TO_DATE('2016-08-22', 'YYYY-MM-DD'), '15');
INSERT INTO ORDERS (ORDERID, ORDERDATE, CUSTOMERID)
VALUES ('1004', TO_DATE('2016-10-22', 'YYYY-MM-DD'), '5');
INSERT INTO ORDERS (ORDERID, ORDERDATE, CUSTOMERID)
VALUES ('1005', TO_DATE('2016-07-24', 'YYYY-MM-DD'), '3');
INSERT INTO ORDERS (ORDERID, ORDERDATE, CUSTOMERID)
VALUES ('1006', TO_DATE('2016-10-24', 'YYYY-MM-DD'), '2');
```

Script Output X

Task completed in 0.469 seconds

1 row inserted.

### OrderLine

OrderID, ProductID, OrderedQuantity, SalePrice

1001, 1, 2  
 1001, 2, 2  
 1001, 4, 1  
 1002, 3, 5  
 1003, 3, 3  
 1004, 6, 2  
 1004, 8, 2  
 1005, 4, 4  
 1006, 4, 1  
 1006, 5, 2

1006, 7, 2  
1007, 1, 3  
1007, 2, 2  
1008, 3, 3  
1008, 8, 3  
1009, 4, 2  
1009, 7, 3  
1010, 8, 10

The screenshot shows the SQL Server Management Studio interface with two panes: a 'Worksheet' pane and a 'Script Output' pane.

**Worksheet Pane:** Contains the following T-SQL script:

```
/* INSERTING IN TABLE ORDERLINE */
INSERT INTO ORDERLINE (ORDERID, PRODUCTID, ORDEREDQUANTITY)
VALUES ('1001', '1', '2');
INSERT INTO ORDERLINE (ORDERID, PRODUCTID, ORDEREDQUANTITY)
VALUES ('1001', '2', '2');
INSERT INTO ORDERLINE (ORDERID, PRODUCTID, ORDEREDQUANTITY)
VALUES ('1001', '4', '1');
INSERT INTO ORDERLINE (ORDERID, PRODUCTID, ORDEREDQUANTITY)
VALUES ('1002', '3', '5');
INSERT INTO ORDERLINE (ORDERID, PRODUCTID, ORDEREDQUANTITY)
VALUES ('1003', '3', '3');
INSERT INTO ORDERLINE (ORDERID, PRODUCTID, ORDEREDQUANTITY)
VALUES ('1004', '6', '2');
```

**Script Output Pane:** Shows the results of the execution:

```
Task completed in 0.39 seconds
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

### PriceUpdate

PriceUpdateID, DateChanged, OldPrice, NewPrice

**3. [42 points]**

The Client wishes that you create the **SQL** code for the following queries (please make sure you **SQL COMMENT** each query and produce the **screenshots for SQL COMMENT, SQL COMMAND, and SQL OUTPUT each, to be placed in Project 2.doc**):

**1. [3 points]**

Which products have a standard price of less than \$ 275?

The screenshot shows a SQL development environment with two main panes. The top pane is titled "Worksheet" and contains the following SQL code:

```
/*Q1 which products have a standard price of less than $275?

PRODUCT NAME          PRODUCT STANDARD PRICE
End Table               175
Coffee Table            200
Computer Desk           250

*/
SELECT P.PRODUCTNAME , P.PRODUCTSTANDARDPRICE
FROM PRODUCT P
WHERE PRODUCTSTANDARDPRICE < 275;
```

The bottom pane is titled "Script Output" and displays the results of the query:

PRODUCTNAME	PRODUCTSTANDARDPRICE
End Table	175
Coffee Table	200
Computer Desk	250

2. (3 points)

List the unit price, product name, and product ID for all products in the Product table.

The screenshot shows the Oracle SQL Developer interface. The top window is titled "Worksheet" and contains a query result. The query is:

```
/* Q2 LIST the unit price, product name, and Product Id for all products in the product tabel
```

UNIT PRICE	PRODUCTNAME	PRODUCT ID
175	End Table	1
200	Coffee Table	2
375	Computer Desk	3
650	Entertainment Center	4
325	Writers Desk	5
750	8-Drawer Desk	6
800	Dining Table	7
250	Computer Desk	8

Below the Worksheet is a "Script Output" window showing the same query results:

UNIT PRICE	PRODUCTNAME	PRODUCT ID
175	End Table	1
200	Coffee Table	2
375	Computer Desk	3
650	Entertainment Center	4
325	Writers Desk	5
750	8-Drawer Desk	6
800	Dining Table	7
250	Computer Desk	8

At the bottom of the Script Output window, it says "8 rows selected."

3. (3 points)

What is the average standard price for all products in inventory?

PS-3380 X

Worksheet Query Builder

```
/*
Q3 What is the average standardprice for all products in inventory?

AVERAGE STANDARD PRICE
440.625
*/
SELECT AVG(P.PRODUCTSTANDARDPRICE) AS "AVERAGE STANDARD PRICE"
FROM PRODUCT P;
```

Script Output X

AVERAGE STANDARD PRICE

-----

440.625

4. (3 points)

How many different items were ordered on order number 1004?

PS-3380 X

Worksheet Query Builder

```
/*
Q4 How many different items were ordered on order number 1004?

ORDERQUANTITY FOR 1004
    4

*/
SELECT SUM(O.ORDERSQUANTITY) AS "ORDERQUANTITY FOR 1004"
FROM ORDERLINE O
WHERE ORDERID = '1004';
```

Script Output X

Task completed in 0.063 seconds

```
ORDERQUANTITY FOR 1004
-----
        4
```

5. (3 points)  
Which orders have been placed since 10/ 24/ 2010?

PS-3380 X

Worksheet Query Builder

```
/*Q5 WHICH ORDERS HAVE BEEN PLACED SINCE 10/24/2010
ORDERID ORDERDATE CUSTOMERID
1002 21-JUL-16 8
1003 22-AUG-16 15
1004 22-OCT-16 5
1005 24-JUL-16 3
1006 24-OCT-16 2
1007 27-AUG-16 5
1008 30-OCT-16 12
1009 05-NOV-16 4
1010 05-NOV-16 1
1001 02-AUG-16 1 */
SELECT *
FROM ORDERS
WHERE ORDERDATE > TO_DATE('10/24/2010', 'MM/DD/YYYY');
```

Script Output X

Task completed in 0.093 seconds

ORDERID	ORDERDATE	CUSTOMERID
1001	21-AUG-16	1
1002	21-JUL-16	8
1003	22-AUG-16	15
1004	22-OCT-16	5
1005	24-JUL-16	3
1006	24-OCT-16	2
1007	27-AUG-16	5
1008	30-OCT-16	12
1009	05-NOV-16	4
1010	05-NOV-16	1

6. (3 points)

What furniture does **COSC3380** carry that isn't made of cherry?

The screenshot shows the SSMS interface with two main windows.

**Query Window:** The title bar says "PS-3380". The toolbar includes icons for Run, Save, Find, and others. The status bar shows "0.078 seconds". The tabs "Worksheet" and "Query Builder" are present. The query text is:

```
Q6 Which furniture does COSC3380 carry that isn't made of cherry?  
FURNITURE  
Coffee Table  
Computer Desk  
Entertainment Center  
8-Drawer Desk  
Dining Table  
Computer Desk  
*/  
  
SELECT P.PRODUCTNAME AS "Furniture"  
FROM PRODUCT P  
WHERE PRODUCTFINISH != 'Cherry';
```

**Output Window:** The title bar says "Script Output". The toolbar includes icons for Run, Save, Find, and others. The status bar shows "Task completed in 0.078 seconds". The output text is:

```
Furniture  
-----  
Coffee Table  
Computer Desk  
Entertainment Center  
8-Drawer Desk  
Dining Table  
Computer Desk  
  
6 rows selected.
```

7. (3 points)

List product name, finish, and standard price for all desks and all tables that cost more than \$ 300 in the Product table.

The screenshot shows the SQL Server Management Studio interface. The top window is titled 'PS-3380' and contains a query in the 'Query Builder' tab:

```
/*
Q7 List product name, finish, and standard price for all desk and all table that costs more than $300 in the Product table
PRODUCTNAME          PRODUCTFINISH          PRODUCTSTANDARDPRICE
-----              -----
Computer Desk        Natural Ash            375
Writers Desk         Cherry                 325
8-Drawer Desk        White Ash             750
Dining Table         Natural Ash            800
*/
SELECT P.PRODUCTNAME, P.PRODUCTFINISH, P.PRODUCTSTANDARDPRICE
FROM PRODUCT P
WHERE (PRODUCTNAME LIKE '%Desk%' OR PRODUCTNAME LIKE '%Table%') AND PRODUCTSTANDARDPRICE > 300 ;
```

The bottom window is titled 'Script Output' and shows the results of the query:

PRODUCTNAME	PRODUCTFINISH	PRODUCTSTANDARDPRICE
Computer Desk	Natural Ash	375
Writers Desk	Cherry	325
8-Drawer Desk	White Ash	750
Dining Table	Natural Ash	800

8. (3 points)

Which products in the Product table have a standard price between \$ 200 and \$ 300?

The screenshot shows the SQL Server Management Studio interface. The top window is titled 'PS-3380' and contains a query in the 'Query Builder' tab:

```
/*
Q8 Which products in the Product table have a standard price between $ 200 and $ 300?
PRODUCTID PRODUCTNAME          PRODUCTFINISH          PRODUCTSTANDARDPRICE PRODUCTLINEID PHOTO
-----  -----
2 Coffee Table        Natural Ash            200                2
8 Computer Desk       Walnut                 250                3
*/
SELECT *
FROM PRODUCT
WHERE PRODUCTSTANDARDPRICE BETWEEN '200' AND '300';
```

The bottom window is titled 'Script Output' and shows the results of the query:

PRODUCTID	PRODUCTNAME	PRODUCTFINISH	PRODUCTSTANDARDPRICE	PRODUCTLINEID	PHOTO
2	Coffee Table	Natural Ash	200	2	
8	Computer Desk	Walnut	250	3	

9. (3 points)

List customer, city, and state for all customers in the Customer table whose address is Florida, Texas, California, or Hawaii. List the customers alphabetically by state and alphabetically by customer within each state.

The screenshot shows the Oracle SQL Developer interface. The top window is titled "PS-3380" and contains a query in the "Worksheet" tab:

```
/*
Q9 List customer, city, and state for all customers in the Customer table whose address is
Florida, Texas, California, or Hawaii. List the customers alphabetically by state and
alphabetically by customer within each state.
CUSTOMERNAME          CUSTOMERCITY      CUSTOMERSTATE
-----
Impressions           Sacramento        CA
Contemporary Casuals Gainesville       FL
Flanigan Furniture   Ft Walton Beach  FL
Value Furnitures     Plano            TX
*/
SELECT C.CUSTOMERNAME, C.CUSTOMERCITY, C.CUSTOMERSTATE
FROM CUSTOMER C
WHERE CUSTOMERSTATE = 'FL' OR CUSTOMERSTATE = 'TX' OR CUSTOMERSTATE = 'CA' OR CUSTOMERSTATE = 'HI'
ORDER BY CUSTOMERSTATE, CUSTOMERNAME;
```

The results are displayed in the "Script Output" tab:

CUSTOMERNAME	CUSTOMERCITY	CUSTOMERSTATE
Impressions	Sacramento	CA
Contemporary Casuals	Gainesville	FL
Flanigan Furniture	Ft Walton Beach	FL
Value Furnitures	Plano	TX

10. (3 points)

Count the number of customers with addresses in each state to which we ship.

PS-3380 X

0.095 seconds

Worksheet Query Builder

```
/*Q10
Count the number of customers with addresses in each state to which we ship.
CUSTOMERSTATE      COUNT(*)
```

CUSTOMERSTATE	COUNT(*)
NJ	1
CA	1
NM	2
VA	1
IL	1
NY	4
CO	1
FL	2
TX	1

```
/*
SELECT C.CUSTOMERSTATE, COUNT(*)
FROM CUSTOMER C
WHERE C.CUSTOMERADDRESS IS NOT NULL
```

Script Output X

Task completed in 0.095 seconds

CUSTOMERSTATE COUNT(\*)

CUSTOMERSTATE	COUNT(*)
NJ	1
CA	1
NM	2
VA	1
IL	1
NY	4
CO	1
FL	2
TX	1

11. (3 points)

Count the number of customers with addresses in each city to which we ship. List the cities by state.

PS-3380 X

Worksheet Query Builder

```

/* Q11
Count the number of customers with addresses in each city to which we ship. List the cities by state.

CUSTOMERSTATE          CUSTMERCITY      COUNT(*)
CA                      Sacramento        1
CO                      Boulder           1
FL                      Ft Walton Beach   1
FL                      Gainesville       1
IL                      Oak Brook         1
NJ                      Carteret          1
NM                      Farmington        1
NM                      Las Cruces        1
NY                      Albany            1
NY                      Rome              2
NY                      Syracuse          1
TX                      Plano             1
VA                      Virginia Beach    1
*/
SELECT DISTINCT C.CUSTOMERSTATE, C.CUSTMERCITY, COUNT(*) AS "TOTAL COUNT"
FROM CUSTOMER C
GROUP BY CUSTOMERSTATE, CUSTMERCITY
ORDER BY CUSTOMERSTATE, CUSTMERCITY;

```

Query Result X

All Rows Fetched: 13 in 0 seconds

	CUSTMERCITY	CUSTOMERSTATE	TOTAL COUNT
1	Sacramento	CA	1
2	Boulder	CO	1
3	Ft Walton Beach	FL	1
4	Gainesville	FL	1
5	Oak Brook	IL	1
6	Carteret	NJ	1
7	Farmington	NM	1
8	Las Cruces	NM	1
9	Albany	NY	1
10	Rome	NY	2
11	Syracuse	NY	1
12	Plano	TX	1
13	Virginia Beach	VA	1

12. (3 points)

Find only states with more than one customer.

PS-3380 X

Worksheet Query Builder

```
/* Q12
Find only states with more than one customer.

CUSTOMERSTATE      TOTAL
-----
NM                  2
NY                  4
FL                  2
*/
```

```
SELECT C.CUSTOMERSTATE, COUNT(*) AS TOTAL
FROM CUSTOMER C
WHERE C.CUSTOMERADDRESS IS NOT NULL
GROUP BY CUSTOMERSTATE
HAVING COUNT(*) >1;
```

Script Output X

Task completed in 0.078 seconds

CUSTOMERSTATE	TOTAL
NM	2
NY	4
FL	2

13. (3 points)

List, in alphabetical order, the product finish and the average standard price for each finish for selected finishes having an average standard price less than 750.

PS-3380 X

Worksheet | Query Builder

```
/* Q13
List, in alphabetical order, the product finish and the average standard price for each finish
for selected finishes having an average standard price less than 750.
```

PRODUCTFINISH	Avg(P.PRODUCTSTANDARDPRICE)
Cherry	250
Natural Ash	458.333333
Natural Maple	650
Walnut	250

```
SELECT P.PRODUCTFINISH, AVG(P.PRODUCTSTANDARDPRICE)
FROM PRODUCT P
GROUP BY PRODUCTFINISH HAVING AVG(P.PRODUCTSTANDARDPRICE)<750
ORDER BY PRODUCTFINISH;
```

Script Output X

Task completed in 0.084 seconds

PRODUCTFINISH	Avg(P.PRODUCTSTANDARDPRICE)
Cherry	250
Natural Ash	458.333333
Natural Maple	650
Walnut	250

14. (3 points)

What is the total value of orders placed for each furniture product?

PS-3380 X

Worksheet Query Builder 0.14 seconds

```
/* Q14
What is the total value of orders placed for each furniture product?

*/
SELECT P.PRODUCTNAME AS "ORDERED PRODUCT", SUM(P.PRODUCTSTANDARDPRICE) AS "TOTAL VALUE"
FROM PRODUCT P, ORDERS O, ORDERLINE OL
WHERE P.PRODUCTID = OL.PRODUCTID AND O.ORDERID = OL.ORDERID
GROUP BY P.PRODUCTNAME;
```

Script Output X

Task completed in 0.14 seconds

ORDERED PRODUCT	TOTAL VALUE
8-Drawer Desk	750
Writers Desk	325
Computer Desk	1875
Coffee Table	400
Entertainment Center	2600
Dining Table	1600
End Table	350

7 rows selected.

**You need to turn in:**

- . The sql scripts with **SQL COMMENTS** for each query copied from this document and the expected output **Project 2.sql. (Only to BB)**

**Example:**

```
# Q1 Which products have a standard price of less than $ 275?  
#   PRODUCTDESCRIPTION      PRODUCTSTANDARDPRICE  
#   End Table                  175  
#   Computer Desk              250  
#   Coffee Table                200
```

- . The **screenshots** of SQL COMMENTS, SQL Oracle COMMANDS and the SQL OUTPUT in Project 2.doc. Please be paper efficient! **(BOTH to BB and HARDCOPY)**

**Without Project 2.doc you will get ZERO!**

Tips for the Projects:

- 1) Prepare your SQL program using the Text editor such as Note Pad **Project 2.sql**.
- 2) Copy and Paste your program from **Project 2.sql** to SQL\*Plus command line.
- 3) If the program works, save the program. Otherwise, fix the error for the program in **Project 2.sql** and try it again.
- 4) Capture screenshots (CROP if needed) of the SQL COMMENTS, SQL COMMAND, and SQL OUTPUT and place them in Project 2.doc.
- 5) Submit **Project 2.sql** that contains all your SQL COMMENTED SQL commands to BB.
- 6) Submit **Project 2.doc** that contains all your screenshots to BB and HARDCOPY.
- 7) Fill in the **Cover Sheet** and you are ready.